Introduction to Security Protocols for Sensor Network

Surender Kumar

***Abstract*— Wireless sensor networks will be widely deployed in the near future. While much research has focused on making these networks feasible and useful, security has received little attention. We present a suite of security protocols optimized for sensor networks: SPINS. SPINS has two secure building blocks: SNEP and TESLA. SNEP includes: data con dentiality, two party data authentication, and evidence of data freshness.. We implemented the above protocols, and show that they are practical even on minimal hardware .the performance of the protocol suite easily matches the data rate of our network. Additionally, we demonstrate that the suite can be used for building higher level protocols.**

***Index Terms*— secure communication protocols, sensor networks, mobile ad hoc networks, MANET, authentication of wireless communication, secrecy and con dentiality, cryptography**

I. INTRODUCTION

We envision a future where thousands to millions of small sensors form self organizing wireless networks. How can we provide security for these sensor networks? Security is not easy; compared with conventional desktop computers, severe challenges exist – these sensors will have limited processing power, storage, bandwidth, and energy.

We need to surmount these challenges, because security is so important. Sensor networks will expand to all aspects of our lives. Here are some typical applications:

Emergency response information: sensor networks will collect information about the status of buildings, people, and transportation pathways. Sensor information must be collected and passed on in meaningful, secure ways to emergency response personnel.

Energy management: in 2001 power blackouts plagued California. Energy distribution will be better managed when we begin to use remote sensors. For example, the power load that can be carried on an electrical line depends on ambient temperature and the immediate temperature on the wire. If these were monitored by remote sensors and the remote sensors received information about desired load and current load, it would be possible to distribute load better. This would avoid circumstances where Californians cannot receive electricity while surplus electricity exists in other parts of the country.

Medical monitoring: we envision a future where individuals with some types of medical conditions receive constant monitoring through sensors that monitor health conditions. For some types of medical conditions, remote sensors may apply remedies (such as instant release of emergency medication to the bloodstream). Logistics and inventory management: commerce in America is based on moving goods, including commodities from locations where surpluses exist to locations where

needs exist. Using remote sensors can substantially improve these mechanisms. These mechanisms will vary in scale – ranging from worldwide distribution of goods through transportation and pipeline networks to inventory management within a single retail store.

Battlefield management: remote sensors can help eliminate some of the confusion associated with combat. They can allow accurate collection of information about current battle eld conditions as well as giving appropriate information to soldiers, weapons, and vehicles in the battled.

This article presents a set of Security Protocols for Sensor Networks, SPINS. The chief contributions of this article are:

Exploring the challenges for security in sensor networks.

Designing and developing TESLA (the "micro" version of TESLA), providing authenticated streaming broadcast.

Designing and developing SNEP (Secure Network Encryption Protocol) providing data con dentiality, two party data authentication, and data freshness, with low overhead.

Designing and developing an authenticated routing protocol using our building blocks.

1.1. Sensor hardware

At UC Berkeley, we are building prototype networks of small sensor devices under the Smart Dust program [45], one of the components of CITRIS. We have deployed these in one of 522

Table 1 Characteristics of prototype SmartDust nodes.

CPU	8bit, 4 MHz
Storage	8 Kbytes instruction ash
	512 bytes RAM
	512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10 Kbps
Operating system	Tiny OS
OS code space	3500 bytes
Available code space	4500 bytes

our EECS buildings, Cory Hall. We are currently using these for a very simple application – heating and air conditioning control in the building. However, the same mechanisms that we describe in this paper can be modified to support sensor that handle emergency system such as re, earthquake, and hazardous material response.

By design, these sensors are inexpensive, low power devices. As a result, they have limited computational and communication resources. The sensors form a self organizing wireless network and form a multi hop routing topology. Typical applications may periodically transmit sensor readings for processing.

Our current prototype consists of nodes, small battery powered devices, that communicate with a more powerful base station, which in turn is connected to an outside network. Table 1 summarizes the performance characteristics of these

Surender Kumar, Guru Jambheswer University Hisar

devices. At 4 MHz, they are slow and underpowered (the CPU has good support for bit and byte level I/O operations, but lacks support for many arithmetic and some logic operations). They are only 8bit processors (note that according to [53], 80% of all microprocessors shipped in 2000 were 4 bit or 8 bit devices). Communication is slow at 10 Kbps.

The operating system is particularly interesting for these devices. We use TinyOS [23]. This small, event driven operating system consumes almost half of 8 Kbytes of instruction ash memory, leaving just 4500 bytes for security and the application.

It is hard to imagine how significantly more powerful devices could be used without consuming large amounts of power. The energy source on our devices is a small battery, so we are stuck with relatively limited computational devices. Wireless communication is the most energy consuming function performed by these devices, so we need to minimize communications overhead. The limited energy supplies create tensions for security: on the one hand, security needs to limit its consumption of processor power; on the other hand, limited power supply limits the lifetime of keys (battery replacement is designed to reinitialize devices and zero out keys).[1]

1.2. Is security on sensors possible?

These constraints make it impractical to use most current secure algorithms, since they were designed for powerful processors. For example, the working memory of a sensor

Base stations differ from nodes in having longer lived energy supplies and additional communications connections to outside networks. PERRIG ET AL.

node is not sufficient to even hold the variables for asymmetric cryptographic algorithms (e.g., RSA [14] with 1024 bits), let alone perform operations with them.

A particular challenge is broadcasting authenticated data to the entire sensor network. Current proposals for authenticated broadcast are impractical for sensor networks. Most proposals rely on asymmetric digital signatures for the authentication, which are impractical for multiple reasons (e.g., long signatures with high communication overhead of 50–1000 bytes per packet, very high overhead to create and verify the signature). Furthermore, previously proposed purely symmetric solutions for broadcast authentication are impractical: Gennaro and Rohatgi's initial work required over 1 Kbyte of authentication information per packet [15], and Rohatgi's improved time signature scheme requires over 300 bytes per packet [14]. Some of the authors of this article have also proposed the authenticated streaming broadcast TESLA protocol [13]. TESLA works well on regular desktop workstations, but uses too much communication and memory on our resource starved sensor nodes. This article extends and adapts TESLA to make it practical for broadcast authentication for sensor networks. We call our new protocol TESLA.

We have implemented all of these primitives. Our measurements show that adding security to a highly resource constrained sensor network is feasible.

Given the severe hardware and energy constraints, we must be careful in the choice of cryptographic primitives and the security protocols in the sensor networks.

II. System assumptions

Before we outline the security requirements and present our security infrastructure, we need to de ne the system architecture and the trust requirements. The goal of this work is to propose a general security infrastructure that is applicable to a variety of sensor networks.

2.1. Communication architecture

Generally, the sensor nodes communicate over a wireless network, so broadcast is the fundamental communication primitive. The baseline protocols account for this property: on one hand they affect the trust assumptions, and on the other they minimize energy usage.

A typical Smart Dust sensor network forms around one or more base stations, which interface the sensor network to the outside network. The sensor nodes establish a routing forest, with a base station at the root of every tree. Periodic transmission of beacons allows nodes to create a routing topology. Each node can forward a message towards a base station, recognize packets addressed to it, and handle message broadcasts. The base station accesses individual nodes using source routing. We assume that the base station has capabilities similar to the network nodes, except that it has sufficient battery power to surpass the lifetime of all sensor nodes, sufficient memory to store cryptographic keys, and means for communicating with outside networks

We do have an advantage with sensor networks, because most communication involves the base station and is not between two local nodes. The communication patterns within our network fall into three categories:

Node to base station communication, e.g., sensor readings.

Base station to node communication, e.g., speci c requests.

Base station to all nodes, e.g., routing beacons, queries or reprogramming of the entire network.

Our security goal is to address these communication patterns, though we also show how to adapt our baseline protocols to other communication patterns, i.e. node to node or node broadcast.

2.2. Trust requirements

Generally, the sensor networks may be deployed in untrusted locations. While it may be possible to guarantee the integrity of the each node through dedicated secure microcontrollers (e.g., [1] or [13]), we feel that such an architecture is too restrictive and does not generalize to the majority of sensor networks. Instead, we assume that individual sensors are untrusted. Our goal is to design the SPINS key setup so a compromise of a node does not spread to other nodes.

Basic wireless communication is not secure. Because it is broadcast, any adversary can eavesdrop on traffic, inject new messages, and replay old messages. Hence, our protocols do not place any trust assumptions on the communication infrastructure, except that messages are delivered to the destination with nonzero probability.

Since the base station is the gateway for the nodes to communicate with the outside world, compromising the base station can render the entire sensor network useless. Thus the base stations are a necessary part of our trusted computing base. Our trust setup reacts this and so all sensor nodes

intimately trust the base station: at creation time, each node gets a master secret key which it shares with the base station. .

Finally, each node trusts itself. This assumption seems necessary to make any forward progress. In particular, we trust the local clock to be accurate, i.e. to have small drift.

2.3. Design guidelines

With the limited computation resources available on our platform, we cannot afford to use asymmetric cryptography and so we use symmetric cryptographic primitives to construct the SPINS protocols. Due to the limited program store, we construct all cryptographic primitives (i.e. encryption, message authentication code (MAC), hash, random number generator) out of a single block cipher for code reuse. To reduce communication overhead we exploit common state between the communicating parties

III. Requirements for sensor network security

This section formalizes the security properties required by sensor networks, and shows how they are directly applicable in a typical sensor network.

3.1. Data confidentiality

A sensor network should not leak sensor readings to neighbouring networks. In many applications (e.g., key distribution) nodes communicate highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving con denasality. Given the observed communication patterns, we set up secure channels between nodes and base stations and later bootstrap other secure channels as necessary.

3.2. Data authentication

Message authentication is important for many applications in sensor networks (including administrative tasks such as network reprogramming or controlling sensor node duty cycle). Since an adversary can easily inject messages, the receiver needs to ensure that data used in any decision making process originates from a trusted source. Informally, data authentication allows a receiver to verify that the data really was sent by the claimed sender. Informally, data authentication allows a receiver to verify that the data really was sent by the claimed sender.

In the two party communication case, data authentication can be achieved through a purely symmetric mechanism: The sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver knows that it must have been sent by the sender.

This style of authentication cannot be applied to a broadcast setting, without placing much stronger trust assumptions on the network nodes. If one sender wants to send authentic data to mutually untrusted receivers, using a symmetric MAC is insecure: any one of the receivers knows the MAC key, and hence, could impersonate the sender and forge messages to other receivers. Hence, we need an asymmetric mechanism to achieve authenticated broadcast. One of our contributions is to construct authenticated broadcast from symmetric primitives only, and introduce asymmetry with delayed key disclosure and one way function key chains.

3.3. Data integrity

In communication, data integrity ensures the receiver that the received data is not altered in transit by an adversary. In SPINS, we achieve data integrity through data authentication, which is a stronger property.

3.4. Data freshness

Sensor networks send measurements over time, so it is not enough to guarantee con denasality and authentication. we also must ensure each message is fresh. Informally, data freshness implies that the data is recent, and it ensures that no adversary replayed old messages. We identify two types of freshness: weak freshness, which provides partial message ordering, but carries no delay information, and strong freshness, which provides a total order on a request–response pair, and allows for delay estimation. Weak freshness is useful for sensor measurements, while strong freshness is useful for time synchronization within the network.

IV. Related work

consider key distribution for resource starved devices in a mobile environment [12]. Park et al. [13] point out weaknesses and improvements. Beller and Yacobi further develop key agreement and authentication protocols [4]. Boyd and Mathuria survey the previous work on key distribution and authentication for resources starved devices in mobile environments [8]. The majority of these approaches rely on asymmetric cryptography. Bergstrom et al. consider the problem of secure remote control of resource starved devices in a home [6].

Fox and Gribble present a security protocol providing secure access to application level proxy services [11]. Their protocol is designed to interact with a proxy to Kerberos and to facilitate porting services relying on Kerberos to wireless devices.

The work of Patel and Crow croft focuses on security solutions for mobile user devices [9]. Unfortunately, their work uses asymmetric cryptography and is, hence, too expensive for the environments we envision.

The work of Czerwinski et al. also relies on asymmetric cryptography for authentication [10].

Stajano and Anderson discuss the issues of bootstrapping security devices [51]. Their solution requires physical contact of the new device with a master device to imprint the trusted and secret information.

Zhou and Haas propose to secure ad hoc networks using asymmetric cryptography [57]. Recently, Basagni et al. proposed to use a network wide symmetric key to secure an ad hoc routing protocol [2]. While this approach is efficient, it does not resist compromise of a single node.

Carman et al. analyze a wide variety of approaches for key agreement and key distribution in sensor networks [9]. They analyze the overhead of these protocols on a variety of hardware platforms.

Marti et al. discuss a mechanism to protect an ad hoc network against misbehaving nodes that fail to forward packets correctly [2]. They propose that each node runs a watchdog (to detect misbehaving neighbouring nodes) and a pathrater (to keep state about the goodness of other nodes); their solution, however, is better suited for traditional networks, with emphasis on reliable point to point

communication, than to sensor networks.

Hubaux et al. present a system for ad hoc peer to peer authentication based on public key certificates [12]. They consider an ad hoc network with nodes powerful enough for performing asymmetric cryptographic operations.

A number of researchers investigate the problem to provide cryptographic services in low end devices. We discuss the hardware efforts, followed by the algorithmic work on cryptography. Several systems integrate cryptographic primitives with low cost microcontrollers. Examples of such systems are secure AVR controllers [1], the Fortezza government standard [15], the Dallas I Button [13], and the Dyad system [15]. These systems support primitives for cryptography, and attempt to zeroes their memory if tampering is devices.

Symmetric encryption algorithms seem to be inherently well suited to low end devices, because they have relatively low overhead. In practice, however, many low end microprocessors are only 4bit or 8bit, and do not provide (efficient) multiplication or variable rotate/shift instructions. Hence many symmetric ciphers are too expensive to implement on our target platform. The Advanced Encryption Standard (AES) [3] Rijndael block cipher [12] is too expensive for our platform. Depending on the implementation, AES was either too big or too slow for our application. Due to our severe limitation on our maximum code size, we chose to use RC5 by Ron Rivest [7]. Algorithms such as TEA by Wheeler and Needham [4] or TREYFER by Yuval [5] would be smaller alternatives, but those other ciphers have not yet been thoroughly analyzed.

V. Conclusion

We designed and built a security subsystem for an extremely limited sensor network platform .We have identified and implemented useful security protocols for sensor networks: authenticated and con denial communication, and authenticated broadcast. We have implemented applications including an authenticated routing scheme and a secure node to node key agreement protocol.

Most of our design is universal and applicable to other networks of low end devices. Our primitives only depend on fast symmetric cryptography, and apply to a wide variety of device configurations. On our limited platform energy spent for security is negligible compared with to energy spent on sending or receiving messages. It is possible to encrypt and authenticate all sensor readings.

The communication costs are also small. Data authentication, freshness, and confi dentiality properties use up a net 6 bytes out of 30 byte packets. So, it is feasible to guarantee these properties on a per packet basis. It is dif cult to improve on this scheme, as transmitting a MAC is fundamental to guaranteeing data authentication.

Certain elements of the design were in fluenced by the available experimental platform. If we had a more powerful platform, we could have used block ciphers other than RC5. The emphasis on code reuse is another property forced by our platform. A more powerful device would allow more modes of authentication. In particular, memory restrictions on buffering limit the effective bandwidth of authenticated broadcast.

Despite the shortcomings of our target platform, we built a system that is secure and works. With our techniques, we believe security systems can become an integral part of practical sensor networks.

VI. Acknowledgements

We gratefully acknowledge funding support for this research. This research was sponsored in part by the United States Postal Service (contract USPS 10259201Z0236), by the United States Defense Advanced Research Projects Agency (contracts DABT6398C0038, “Ninja”, N660019928913, “Endeavour”, and F3361501C1895, “NEST”), by the United States National Science Foundation (grants FD9979852 and RI EIA9802069) and from gifts and grants from the California MICRO program, Intel Corporation, IBM, Sun Microsystems, and Philips Electronics. DARPA Contract N660019928913 is under the supervision of the Space and Naval Warfare Systems Center, San Diego. This paper represents the opinions of the authors and do not necessarily represent the opinions or policies, either expressed or implied, of the United States government, of DARPA, NSF, USPS, or any other of its agencies, or any of the other funding sponsors.

We thank Jean Pierre Hubaux, Dawn Song and David Wagner for helpful discussions and comments. An earlier version of this work appeared as [44].

References

[1] Atmel, Secure Microcontrollers for SmartCards, http://www. atmel.com/atmel/acrobat/1065s.pdf

[2] S. Basagni, K. Herrin, E. Rosti and D. Bruschi, Secure Pebblenets, in: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001) (2001) pp. 156–163.

[3] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation, in: Symposium on Foundations of Computer Science (FOCS) (1997).

[4] M. Beller and Y. Yacobi, Fully edged twoway public key authenti cation and key agreement for lowcost terminals, Electronics Letters 29(11) (1993) 999–1001.

[5] S. Bellovin and M. Merrit, Augmented encrypted key exchange: a passwordbased protocol secure against dictionary attacks and password le compromise, in: ACM Conference on Computer and Communications Security CCS1 (1993) pp. 244–250.

[6] P. Bergstrom, K. Driscoll and J. Kimball, Making home automation communications secure, IEEE Computer 34(10) (2001) 50–56.

[7] A. Biryukov and D. Wagner, Slide attacks, in: International Workshop on Fast Software Encryption (1999).

[8] C. Boyd and A. Mathuria, Key establishment protocols for secure mobile communications: A selective survey, in: Australasian Conference on Information Security and Privacy (1998) pp. 344–355.

[9] D.W. Carman, P.S. Kruus and B.J. Matt, Constraints and approaches for distributed sensor network security, NAI Labs Technical Report No. 00010 (2002).

[10] S.E. Czerwinski, B.Y. Zhao, T.D. Hodes, A.D. Joseph and R.H. Katz, An architecture for a secure service discovery service, in: ACM International Conference on Mobile Computing and Networking (MobiComÕ99) (1999) pp. 24–35.

[11] D. Johnson, D.A. Maltz and J. Broch, The dynamic source routing protocol for mobile ad hoc networks, Internet draft, Mobile AdHoc Network (MANET) Working Group, IETF (1999).

[12] J. Daemen and V. Rijmen, AES proposal: Rijndael (1999).

[13] Dallas, iButton: A Javapowered cryptographic iButton, http:// www.ibutton.com/ibuttons/java.html

[14] W. Dif e and M.E. Hellman, Privacy and authentication: An introduction to cryptography, Proceedings of the IEEE 67(3) (1979) 397–427.

[15] Fortezza, Fortezza: Application implementers guide (1995).